IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s):	David Stanton		
Assignee:	Hewlett Packard Company , L.P.		
Title:	**SYSTEM AND METHOD FOR COMPONENT-BASED SOFTWARE DEVELOPMENT**		
Serial No.:	09/616,330	Filing Date:	July 15, 2000
Examiner:	Phan, Tam T.	Group Art Unit:	2144
Docket No.:	200407537-1	KB Ref. No.	1015.P132US

Irvine, California
August 15, 2005

MAIL STOP APPEAL BRIEFS - PATENTS
COMMISSIONER FOR PATENTS
P.O. BOX 1450
ALEXANDRIA, VA 22313-1450

APPELLANT'S BRIEF

Dear Sir:

This paper is responsive to the Final Office Action dated March 16, 2005, having a shortened statutory period expiring June 16, 2005. Reconsideration is respectfully requested.

I. REAL PARTY IN INTEREST

The entire interest in the present application has been assigned to Hewlett-Packard Company LP, a Texas Limited Partnership having a place of business at 20555 S.H. 249, Houston, Texas, 77070, as recorded at reel 016056, frame 0656.

II. RELATED APPEALS AND INTERFERENCES

No other appeals or interferences are known to the appellant, the appellant's legal representative, or assignee which will directly affect or be directly affected by or have bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 1-12 are canceled.

Claims 13-36 are pending in the application. Claims 13-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kumar *et al.* (U.S. Patent Number 6,542,515), hereinafter referred to as Kumar, in view of Box *et al.* (Simple Object Access Protocol (SOAP) 1.1; May 2000).

Claims 22-36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ankireddipally *et al.* (U.S. Patent Number 6,772,216), hereinafter referred to as Ankireddipally, in view of Young (U.S. Patent Number 6,560,606), hereinafter referred to as Young.

The rejections of Claims 13 - 36 are on appeal.

IV. STATUS OF AMENDMENTS

The appellant's reply dated May 23, 2005, in response to the final office action dated March 16, 2005, was entered.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Independent Claim 13 pertains to a messaging platform (105, 25) for a component-based software system. The platform includes a connection assembler (25) for at least one of creating, managing, and manipulating a first messaging platform connection. The platform further includes a protocol management framework (25) for implementation of a predetermined transport protocol over the first messaging platform connection; and a schema generator (25) for, responsive to a request for service received over a second messaging platform connection, creating a document according to a predetermined format, the document containing information to be provided to another system over the first messaging platform connection. The platform further includes an encoding component (60) for converting a document in the predetermined format into a first encoded object that can be understood and used by the other system, the first encoded object being encoded according to a default encoding protocol; and a translation component for encoding a document in the predetermined format into a second encoded object that can be understood and used by the other system. The second encoded object being encoded according to an encoding protocol different from the default encoding protocol. The elements in Claim 13 are described at least on pages 21 line 3 through page 25 line 16 of the specification, and are shown at least in Figures 1, 9, 10.

Independent Claim 22 pertain to an apparatus including logic instructions operable to receive a service request (50) from a sender, wherein the service request invokes a service component (45) and provides parameters required by the service component. The logic instructions are further operable to determine whether the service component invoked by the service request is available; determine the parameters in the service request that are required by the service component; create a request document (35) that includes the parameters required by the service component based on at least some of the parameters in the service request; create an encoder object (Application Appendix I, page 22, Part One: Item 2) upon receipt of the service request, wherein the encoder object identifies a handler (Application Appendix I, page 23, Part Two: Item 2) that translates the request document to a document format required by the service component; and transmit the encoder object and the request document to a system hosting the service component.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Whether independent claim 13 and corresponding dependent claims 14-21 are unpatentable under 35 U.S.C. §103(a) over Kumar *et al.* (U.S. Patent Number 6,542,515), hereinafter referred to as Kumar, in view of Box *et al.* (Simple Object Access Protocol (SOAP) 1.1; May 2000) hereinafter referred to as Box, taking into account all limitations of the claims.

2. Whether independent claim 22 and corresponding dependent claims 23-36 are unpatentable under 35 U.S.C. §103(a) over Ankireddipally *et al.* (U.S. Patent Number 6,772,216), hereinafter referred to as Ankireddipally, in view of Young (U.S. Patent Number 6,560,606), hereinafter referred to as Young, taking into account all limitations of the claims.

VII. ARGUMENT

1. Rejection of Claims 13-21 under 35 U.S.C. §103

Claims 13-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kumar *et al.* (U.S. Patent Number 6,542,515), hereinafter referred to as Kumar, in view of Box *et al.* (Simple Object Access Protocol (SOAP) 1.1; May 2000). To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. MPEP § 2143. Failure to meet just one of the three prongs for the test of obviousness is sufficient to defeat rejection of the claims under 103(a).

The applicants appeal the rejection of independent Claim 13 because there would be no reasonable expectation of success in combining the teachings of Kumar and Box as suggested by the Examiner. Kumar discloses a combination of XML document type definition (DTD) documents transmitted via HTTP, and an adapter to unpack stacked request messages in the XML DTD for use in an Application Programming Interface (API). (Kumar, FIGs. 6 and 7, and col. 14 line 28 through col. 15 line 35.) Box describes a SOAP message is an XML document that consists of a mandatory SOAP envelope, an optional SOAP Header, and a mandatory SOAP Body. (Box *et al.*, SOAP v. 1.1, W3C Note 08 May 2000, Section 1). Notably, a SOAP message must not contain a Document Type Declaration (DTD), as indicated in Section 3 of the SOAP Specification v 1.1 as follows:

"**3. Relation to XML**

All SOAP messages are encoded using XML (see [7] for more information on XML).

A SOAP application SHOULD include the proper SOAP namespace on all elements and attributes defined by SOAP in messages that it generates. A SOAP application MUST be able to process SOAP namespaces in messages that it receives. It MUST discard messages that have incorrect namespaces (see section 4.4) and it MAY process SOAP messages without SOAP namespaces as though they had the correct SOAP namespaces.

SOAP defines two namespaces (see [8] for more information on XML namespaces):

- The SOAP envelope has the namespace identifier "http://schemas.xmlsoap.org/soap/envelope/"
- The SOAP serialization has the namespace identifier "http://schemas.xmlsoap.org/soap/encoding/"

A SOAP message MUST NOT contain a Document Type Declaration. A SOAP message MUST NOT contain Processing Instructions. [7]

SOAP uses the local, unqualified "id" attribute of type "ID" to specify the unique identifier of an encoded element. SOAP uses the local, unqualified attribute "href" of type "uri-reference" to specify a reference to that value, in a manner conforming to the XML Specification [7], XML Schema Specification [11], and XML Linking Language Specification [9].

With the exception of the SOAP mustUnderstand attribute (see section 4.2.3) and the SOAP actor attribute (see section 4.2.2), it is generally permissible to have attributes and their values appear in XML instances or alternatively in schemas, with equal effect. That is, declaration in a DTD or schema with a default or fixed value is semantically equivalent to appearance in an instance."

Box *et al.*, SOAP v. 1.1, W3C Note 08 May 2000, Section 3, emphasis added.

The Examiner asserts that it would have been obvious to modify the messaging platform of Kumar with the teachings of Box to exchange information in a distributed environment, and that SOAP defines a simple mechanism for expressing application semantics for encoding data within modules. (Final Office Action, paragraph 12). In Paragraph 44 of the Final Office Action, the Examiner quotes the portion of the Box reference that states "it is generally permissible to have attributes and their values appear in XML instances or alternatively in schemas, with equal effect. That is, <u>declaration in a DTD or schema with a default or fixed value is semantically equivalent to appearance in an instance</u>." (Emphasis added).

With respect, Applicant disagrees. The cited portion of Box states that declaration in a DTD or schema with a default or fixed value is semantically equivalent to appearance in an instance. The cited portion of Box DOES NOT state that DTD can be used in a SOAP message, but rather that a SOAP message MUST NOT contain a DTD. Semantic equivalence in a programming language does not mean that the syntax required to implement the feature will be accepted by a compiler for the programming language or that it will not generate a runtime error upon execution, however. Accordingly, the syntax for a DTD is not interchangeable with the syntax for an XML schema or instance, despite semantic equivalence.

Notably, the follow-on recommendations for SOAP v1.2 state the following:

"5. SOAP Message Construct

A SOAP message is specified as an XML infoset that consists of a *document information item* with exactly one member in its [children] property, which MUST be the SOAP `Envelope` *element information item* (see **5.1 SOAP Envelope**). This *element information item* is also the value of the [document element] property. The [notations] and [unparsed entities] properties are both empty. The [base URI], [character encoding scheme] and [version] properties can have any legal value. The [standalone] property either has a value of "yes" or has no value.

<u>The XML infoset of a SOAP message MUST NOT contain a *document type declaration information item*.</u>"

Gudgin *et al.*, SOAP v. 1.2, W3C Recommendation 24 June 2003, Section 5, emphasis added. Note that Gudgin does not include the paragraph in Box that states "it is generally permissible to have attributes and their values appear in XML instances or alternatively in schemas, with equal effect…".

Thus, Applicant asserts that DTDs are not allowed in SOAP messages, and the combination of Kumar and Box would have no reasonable expectation of success. Claim 13 is distinguishable from Kumar and Box, alone and in combination, for at least the above-mentioned reasons. Claims 14-21 depend from Claim 13 and include features that further distinguish them from the prior art. Allowance of Claims 13-21 is respectfully requested.

2. Rejection of Claims 22-36 under 35 U.S.C. §103

Claims 22-36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ankireddipally in view of Young (U.S. Patent Number 6,560,606).

Claim 22 recites:

"An apparatus comprising:
logic instructions operable to:
...
create an encoder object upon receipt of the service request, wherein the encoder object identifies a handler that translates the request document to a document format required by the service component; and
transmit the encoder object and the request document to a system hosting the service component."

Ankireddipally is cited as suggesting such features, and Young is cited as teaching these features. The cited portions of Ankireddipally pertain to encoding, translating, and interpreting input and output parameter data types; sending a request to perform a service including data for arguments required to perform the service, and boilerplate language at the end of the detailed description stating that the claims are not limited to the specific embodiments of the description. (Ankireddipally, col. 17 lines 23-30, col. 18 lines 9-23, col. 26 lines 9-17). The cited portion of Young pertains to an application programming interface (API) that reformats data as required, an object generator that generates session objects containing properties with values representing the usage data, and a transmission module that includes a serializer, an encoder, and a transmitter. The encoder encodes the object stream for error detection purposes and/or authentication purposes. (Young, col. 6 lines 46-67).

Nothing in Ankireddipally or Young, alone or in combination, discloses or suggests an encoder object that identifies a handler that translates the request document to a document format required by the service component, or transmits the encoder object and the request document to a system hosting the service component, however, as set forth in Claim 22. The Examiner admits that Ankireddipally does not disclose or suggest an encoder object or transmitting the encoder object and the request document. (Final Office Action paragraph 24). The session objects in Young contain properties with values for the usage data. Further, the encoder in Young encodes the stream of session objects for error detection or authentication purposes. There is no feature in Young that is used to identify a handler or translate the request document to a document format required by the service component, as required by Claim 22. Claim 22 is distinguishable from Ankireddipally and

Young, alone and in combination, for at least these reasons.

Claims 23 - 36 depend from Claim 22 and include features that further distinguish them from the cited references. Allowance of Claims 22 -36 is respectfully requested.

I hereby certify that this correspondence is being facsimile transmitted to the USPTO at (571)273-8300 on the date shown below:

[illegible]
(Signature)

Mary Jo Bertani
(Printed Name of Person Signing Certificate)

August 15, 2005
(Date)

Respectfully submitted,

[illegible]

Mary Jo Bertani
Attorney for Applicant(s)
Reg. No. 42,321

VIII. CLAIMS APPENDIX

Claims remaining in the application are as follows:

Claims 1-12 (Canceled)

13. (Previously Presented) A messaging platform for a component-based software system, the platform comprising:

a connection assembler for at least one of creating, managing and manipulating a first messaging platform connection;

a protocol management framework for implementation of a predetermined transport protocol over the first messaging platform connection;

a schema generator for, responsive to a request for service received over a second messaging platform connection, creating a document according to a predetermined format, the document containing information to be provided to another system over the first messaging platform connection;

an encoding component for converting a document in the predetermined format into a first encoded object that can be understood and used by the other system, the first encoded object being encoded according to a default encoding protocol; and

a translation component for encoding a document in the predetermined format into a second encoded object that can be understood and used by the other system, the second encoded object being encoded according to an encoding protocol different from the default encoding protocol.

14. (Original) The platform of claim 13, wherein:

the information is provided by a service component; and

the request for service is in a form not understandable by the service component.

15. (Original) The platform of claim 13, wherein the service request is in a platform and application-independent format.

16. (Original) The platform of claim 15, wherein the service request is in an Extensible Markup Language format.

17. (Original) The platform of claim 13, further comprising a lookup service for determining a service component to handle the service request.

18. (Original) The platform of claim 17, wherein the lookup service determines the service component to handle the service request based on information associated with the service component.

19. (Original) The platform of claim 13, wherein the protocol management framework implements HTTP for transport.

20. (Original) The platform of claim 13, wherein the default protocol is SOAP.

21. (Previously Presented) The platform of claim 13, wherein an identifier of the encoding object is included in a Universal Resource Locator (URL), and the URL is sent to the second messaging platform in combination with the document in the predetermined format.

22. (Previously Presented) An apparatus comprising:
logic instructions operable to:
receive a service request from a sender, wherein the service request invokes a service component and provides parameters required by the service component;
determine whether the service component invoked by the service request is available;
determine the parameters in the service request that are required by the service component;
create a request document that includes the parameters required by the service component based on at least some of the parameters in the service request;
create an encoder object upon receipt of the service request, wherein the encoder object identifies a handler that translates the request document to a document format required by the service component; and
transmit the encoder object and the request document to a system hosting the service component.

23. (Previously Presented) The apparatus of Claim 22 further comprising:
logic instructions operable to:
invoke the service component using the second document;
receive a response from the service component in the document format required by the service component;
convert the response to a response message in a platform-independent format;
provide the response message to a message platform;
convert the response message to a format required by the sender.

24. (Previously Presented) The apparatus of Claim 22, wherein the encoding object is included in a Universal Resource Locator (URL).

25. (Previously Presented) The apparatus of Claim 22 further comprising:
logic instructions operable to:
create a connection information file that identifies an external host on which the service component resides.

26. (Previously Presented) The apparatus of Claim 25 wherein the connection information file further identifies a port on which the service component resides.

27. (Previously Presented) The apparatus of Claim 25 wherein the connection information file further identifies a transport mode and a handler that is operable to create the second document.

28. (Previously Presented) The apparatus of Claim 22 further comprising:
logic instructions operable to:
generate a Graphical User Interface (GUI) to identify and store addresses of external host computers on which service components that can be invoked by the sender reside.

29. (Previously Presented) The apparatus of Claim 22 further comprising:
logic instructions operable to:
generate a repository of information regarding the service component, wherein the repository includes at least one of the group consisting of: an identifier, a classification, dependencies, and version of the service component.

30. (Previously Presented) The apparatus of Claim 29 wherein the repository includes a component data section and a shared data section, and the shared data section includes information that can be shared between service components and changed dynamically at runtime.

31. (Previously Presented) The apparatus of Claim 22 further comprising:
logic instructions operable to:
create a catalog of components in the repository, wherein the catalog can be accessed by authorized users.

32. (Previously Presented) The apparatus of Claim 22 further comprising:
logic instructions operable to:
generate instructions to specify remote interfaces and provide an interface for writing remote objects.

33. (Previously Presented) The apparatus of Claim 29 further comprising:
logic instructions operable to:
map relationships between objects in the service components.

34. (Previously Presented) The apparatus of Claim 22 further comprising:
logic instructions operable to:
implement a transaction protocol to coordinate transactions from disparate systems communicated via different protocols.

35. (Previously Presented) The apparatus of Claim 22 further comprising:
logic instructions operable to:
select an alternate service component when the service component invoked by the service request is not available.

36. (Previously Presented) The apparatus of Claim 35 wherein further comprising:
logic instructions operable to:
select the alternate service component based on at least one factor of the group consisting of: queue depth, average compute time, and network latency.

XI. EVIDENCE APPENDIX

Part A of this Appendix includes the pertinent excerpt from Box *et al.*, SOAP v. 1.1, W3C Note 08 May 2000, Section 3.

Part B of this Appendix includes the pertinent excerpt from Gudgin *et al.*, SOAP v. 1.2, W3C Recommendation 24 June 2003, Section 5.

XI. Evidence Appendix

Part A